VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

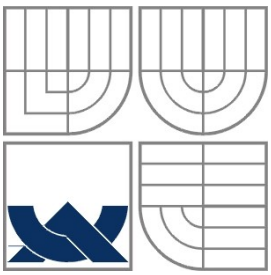
APLIKACE PRO VÝUKU 2D KŘIVEK

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

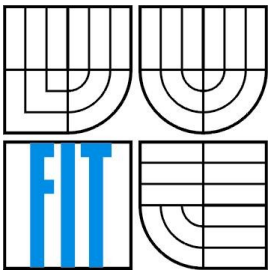
AUTOR PRÁCE
AUTHOR

PAVEL OPLETAL

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

APLIKACE PRO VÝUKU 2D KŘIVEK

APPLICATION FOR 2D CURVES DEMONSTRATION

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — PAVEL OPLETAL
AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ VENERA
SUPERVISOR

BRNO 2008

Abstrakt

Tato bakalářská práce se zabývá dvojrozměrnými křivkami používanými v počítačové grafice. Shrnuje obecnou problematiku těchto křivek a dále se zabývá konkrétními metodami pro jejich výpočet. Popisuje metody užívané pro výpočet Fergusonových kubik, křivek Kochanek-Bartels, Kardinálního splinu, Catmull-Rom splinu, Bézierových křivek a jejich modifikací, Coonsových kubik, Coonsových kubických B-splinů a křivek NURBS. Praktická část této práce se zabývá návrhem a implementací výukové aplikace, která demonstruje vybrané křivky.

Klíčová slova

křivky, interpolační křivky, aproximační křivky, Ferguson, Catmull-Rom, Bézierovy křivky, algoritmus de Casteljau, Coonsovy kubiky, NURBS, demonstrační aplikace, .NET, C#

Abstract

This bachelor's thesis deals with planar curves used in computer graphics. It sums up general ideas of these curves and deals with particular methods for computing planar curves. It describes methods used for computing Ferguson cubic curves, Kochanek-Bartels spline, Cardinal spline, Catmull-Rom spline, Bézier curves and their modifications, Coons cubic curves, Coons cubic B-spline curves and NURBS. Practical part of this project deals with concept and impementation of tutorial, which demonstrates chosen curves.

Keywords

curves, interpolating curves, approximating curves, Ferguson, Catmull-Rom, Bézier curves, de Casteljau's algorithm, Coons cubic, Coons cubic B-spline, NURBS, tutorial, .NET, C#

Citace

Pavel Opletal: Aplikace pro výuku 2D křivek, bakalářská práce, Brno, FIT VUT v Brně, 2008

Aplikace pro výuku 2D křivek

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jiřího Venery. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Pavel Opletal
20.7.2008

Poděkování

Rád bych tímto poděkoval Ing. Jiřímu Venerovi za jeho vedení, věcné připomínky a rady, které mi pomohly při vytváření této práce.

Obsah

1 Úvod

Počítačová grafika je odvětví informatiky, které v posledních letech získalo na důležitosti. Díky rozvoji výpočetní techniky se můžeme s počítačovou grafikou, především dvojrozměrnou, setkat téměř na každém kroku. Za pomoci různých počítačových nástrojů jsou dnes vytvářeny knihy, noviny, či plakáty. Prostředky dvojrozměrné počítačové grafiky jsou také využívány při tvorbě webových stránek, prezentací a stále intenzivněji pro zpracování digitálních fotografií nebo videí. Neméně důležitá je ale i pro tvorbu uživatelských rozhraní aplikací.

Jelikož se ovšem v našem světě příliš často nevyskytuje ideál rovnosti přímek a kulatosti kružnic, je nutné, aby zejména odborníci zabývající se počítačovou grafikou uměli popsat křivé tvary. Pochopení této problematiky není vždy zcela jednoduché. Smyslem této práce je nastudovat metody a algoritmy používané pro výpočet počítačových křivek a následně vytvořit výukovou aplikaci, která napomůže případným zájemcům pochopit principy a metody používané pro výpočet a následné zobrazení křivek v počítačích.

Začátek této práce je věnován obecné problematice počítačových křivek. Kapitola rozebírá vyjádření křivek, jejich vlastnosti a modelování. Následující kapitola se zabývá interpolačními křivkami, konkrétně metodami pro výpočet Fergusonových kubik, křivek Kochanek-Bartels, kardinálním splinem a Catmull-Rom splinem. Další kapitola je věnována problematice aproximačních křivek. Jsou zde vysvětleny principy metod pro výpočet Bézierových křivek, aloritmus de Cateljau a racionálních Bézierových křivek. Dále se kapitola zabývá Coonsovými kubikami a křivkami NURBS. Poslední část se věnuje použitým prostředkům při vývoji, návrhu uživatelského rozhraní a dalším krokům, které vedly k úspěšnému dokončení aplikace.

Při tvorbě textu zabývajícím se teorií počítačových křivek jsem často čerpal z knihy Moderní počítačová grafika [1], některé informace byly převzaty i ze skript ČVUT Algoritmy počítačové grafiky [2]. Mnoho jich ovšem není, protože je již tato publikace na dnešní dobu poměrně zastaralá. Zbylé zdroje jsou uvedeny přímo u částí práce, kterých se to týká.

2 Křivky

Jak již bylo nastíněno v úvodu, křivky mají v počítačové grafice a v souvisejících aplikacích velké využití. Tato oblast je poměrně široká, protože různé aplikace mají i různé požadavky. S křivkami se můžeme setkat ve dvou i ve třech dimenzích. Užívají se například pro definici fontů, pro určení dráhy pohybu objektu nebo pro šablonování. Křivky jsou nejlépe reprezentovány funkcemi, což je ovšem v počítačové grafice problematické. Formulace funkce ve tvaru $y = f(x_{1,} x_{2,} \ldots, x_n)$, která by popsala přesný tvar křivky, je na základě představy o tvaru velmi obtížná a často také nemožná. V případě použití běžně známých funkcí je zase problematické nastavení parametrů tak, aby byly splněny požadované podmínky. Proto se tedy hledají metody, které by umožňovaly co nejjednodušší zadávání křivek a pokud možno, aby se dal i jejich tvar odhadnout dopředu. Většinou jde o to, že uživatel zadá jen několik bodů, kterým se v této oblasti říká řídící body, a matematický aparát se sám postará o vypočtení křivky.

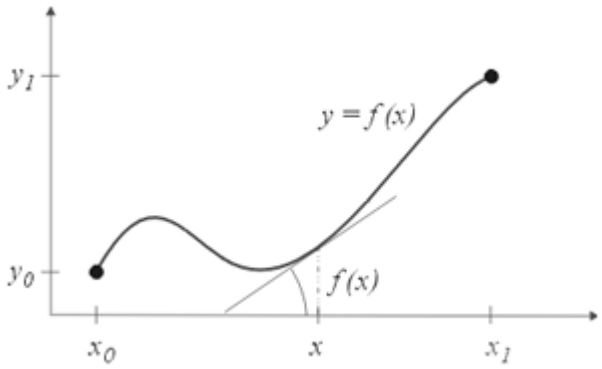
K rozvoji tohoto přístupu přispěl například i rozvoj automobilového průmyslu. V roce 1959 používal u firmy Citroen Paul de Casteljau matematický model křivek a ploch, jenž mu umožňoval jejich jednoduché zadávání. Podobně u konkurenční automobilky Renault vedl v šedesátých letech vývoj programového systému UNISURF Pierre Bézier. V sedmdesátých letech si tato disciplína zahrnující návrh, zobrazování a analýzu křivek a ploch vysloužila dokonce vlastní název, CAGD (Computer Aided Geometry Design). Metody CAGD se postupem času zdokonalovaly. V současné době je k dispozici velmi silný nástroj, který je stále aktivně rozvíjen.

Výrazný pokrok do této oblasti vneslo používání racionálních B-Spline křivek a ploch s neuniformní parametrizací, NURBS (Non-uniform Rational B-Spline). Důležitou vlastností těchto metod je schopnost generovat klasické geometrické prvky, jako jsou koule, válec, kružnice atp., za pomoci metod aproximace.

2.1 Vyjádření a vlastnosti křivek

Křivky bývají v počítačích obvykle reprezentovány jako soustava parametrů nějaké rovnice. Ta je posléze generativně zobrazována. Toto vyjádření může být v podstatě trojího druhu – explicitní, implicitní a parametrické.

Explicitní vyjádření křivky může být zadáno například jako spojitá funkce ve tvaru $y = f(x)$. Křivka bývá orientována ve směru rostoucího *x* (obr. 2.1). Toto zadání křivky lze ovšem využít pouze u křivek, které jsou zároveň funkcemi, tzn. že každé hodnotě *x* z definičního oboru odpovídá jen jedna jediná funkční hodnota *y*.



Obr. 2.1: Explicitně vyjádřená křivka

Implicitní zadání křivky má tvar $F(x, y) = 0$. Toto zadání není v mnoha případech vhodné v porovnání s ostatními. V obecnějších případech totiž neumožňuje postupný výpočet křivky. Význam toto zadání má například při výpočtu průsečíků se zadanou křivkou nebo při testování oblastí vymezených touto křivkou.

Třetí možností vyjádření křivek je parametrické vyjádření. To je v počítačové grafice využíváno nejčastěji. Z fyzikálního hlediska lze křivku chápat jako dráhu bodu, který se po ní pohybuje v čase *t*. Lze tedy říci, že souřadnice bodu jsou funkcemi parametru *t*. Parametrické vyjádření křivek má tvar

$$x = x(t),\ y = y(t).$$

Uvedený tvar je určen pro výpočet křivky v rovině. Pokud bychom chtěli vyjádřit křivku v prostoru, stačilo by pouze přidat příslušnou rovnici pro třetí rozměr. Parametr *t* je z intervalu $t \in \langle t_{min}, t_{max} \rangle$ a nejčastěji bývá volen v rozsahu $\langle 0,1 \rangle$. Parametrickými funkcemi je určena bodová rovnice křivky

$$Q(t) = [x(t), y(t)],$$

nebo vektorová rovnice

$$\vec{q}(t) = (x(t), y(t)).$$

Vektor $\vec{q}(t) = Q(t) - [0,0]$ se nazývá polohový vektor a jeho velikost je rovna vzdálenosti bodu $Q(t)$ od počátku. Výhodou parametrického zápisu je závislost výsledných souřadnic na jediném parametru, jehož fyzikální interpretací je čas. Díky tomu lze vyjádřit průběhy, kdy křivka prochází vícekrát stejnými body v rovině, tzn. může se křížit, uzavřít apod.

U parametrických křivek je důležitý také tečný vektor. Jeho hlavní význam bude uveden níže. Tečný vektor v bodě $Q(t_0)$ je určen derivacemi parametricky vyjádřené křivky po složkách ve tvaru

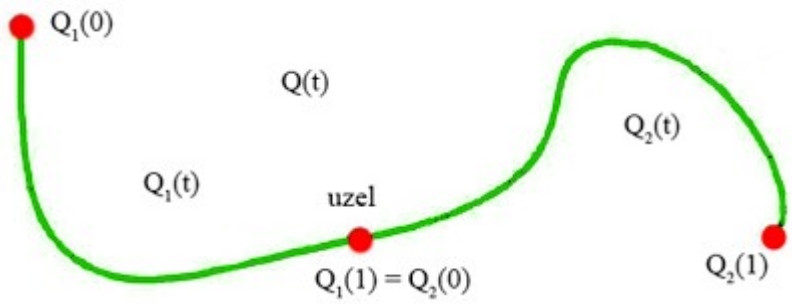
$$\vec{q}\,'(t_0) = (x'(t_0), y'(t_0)) = \left(\frac{dx(t_0)}{dt}, \frac{dy(t_0)}{dt} \right).$$

Rovnice tečny se dá z tečného vektoru a bodu na křivce vypočítat jako

$$P(u) = Q(t_0) + u\vec{q}\,'(t_0).$$

Vektor $\vec{q}'(t)$ se také nazývá směrový vektor přímky. Z předchozích vztahů je patrné, že křivka vyjádřená parametricky umožňuje snadno vyjádřit tečny křivky. Toho se dá s výhodou využít zejména pro navazování křivek a skládání složitějších tvarů z křivek jednodušších.

V tomto místě se tedy dostáváme k tomu, proč je tečný vektor tak důležitý. V praxi se totiž se skládáním jednodušších segmentů do výsledné křivky setkáváme poměrně často. Důležitý je způsob jejich napojení, zejména tzv. spojitost v bodech napojení křivek. Předpokládejme, že $Q_1(t)$ a $Q_2(t)$ jsou dvě části křivky $Q(t)$ spojené v bodě $Q_1(1)=Q_2(0)$ (viz obr. 2.2). Bod, ve kterém se křivky stýkají, je také někdy nazýván uzel.



Obr. 2.2: Křivka $Q(t)$ vzniklá spojením segmentů $Q_1(t)$ a $Q_2(t)$

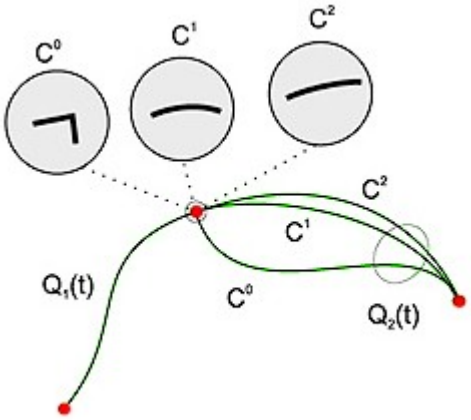
Obecně platí, že křivka je spojitá, pokud je spojitá ve všech svých bodech, zejména pak v uzlech. Pokud požadujeme, aby byla křivka hladká, musí platit, že křivka musí mít ve všech svých bodech spojité navíc i první derivace. U vyšších derivací obecně říkáme, že je křivka *n*-tého řádu, kde *n* je stupeň derivace.

U křivek v počítačové grafice rozlišujeme dva základní druhy spojitosti v uzlech. První se nazývá parametrická spojitost a označuje se C^n, kde *n* je stupeň spojitosti. Říkáme, že křivka $Q(t)$ je třídy C^n, má-li ve všech bodech spojité derivace podle parametru *t* do řádu *n*. Spojitost třídy C^0 znamená, že dva segmenty křivky jsou spojitě navázány tak, že koncový bod prvního segmentu křivky je zároveň počátečním bodem druhého segmentu. Spojení křivek třídy C^1 lze dosáhnout, pokud zajistíme rovnost tečných vektorů v koncovém bodě segmentu Q_1 a počátečním bodě segmentu Q_2. Pro C^2 je analogicky požadována rovnost vektoru první a druhé derivace (obr 2.3), atd. Zkráceně lze vztah mezi vektory koncového bodu prvního segmentu a počátečního bodu druhého segmentu zapsat

$$\vec{q_1^{(i)}}(1) = \vec{q_2^{(i)}}(0); \quad \forall\, i = 0,1,\ldots,n\,.$$

Čím je spojitost vyšší, tím déle se oba segmenty přimykají ke stejnému směru. Ze spojitosti C^0 plyne, že je dráha pohybu bodu po křivce spojitá. V uzlu ovšem může bod změnit skokem směr

pohybu, rychlost i zrychlení. Směr pohybu a velikost rychlosti se při spojitosti C^1 měnit skokem nemůže a při spojitosti C^2 zůstává nezměněné i zrychlení.

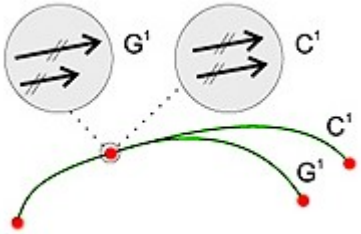


Obr. 2.3: Parametrická spojitost C^0, C^1 a C^2

Dalším druhem spojitosti je tzv. geometrická spojitost označovaná G^n. Nejčastěji se používá geometrická spojitost G^0 a G^1. Pro spojitost G^0 opět platí, že je křivka spojitá, pokud koncový bod prvního segmentu je totožný s počátečním bodem druhého segmentu. Pro spojitost G^1 ovšem platí, že dva segmenty jsou spojité, pokud jsou G^0 spojité a zároveň tečné vektory obou segmentů jsou lineárně závislé (obr. 2.4). Platí tedy vztah

$$\vec{q}\,'_1(1) = k\,\vec{q}\,'_2(0); \quad k > 0 .$$

Tato spojitost zaručuje totožnost tečen, nikoliv tečných vektorů. Pohybující se bod nemůže skokem změnit směr, ale rychlost ano. Ze subjektivního hlediska zaručuje spojitost G^1 téměř shodnou hladkost spojení jako C^1. V praxi bývá daleko snazší zaručit spojitost G^1 než C^1.



Obr. 2.4: Geometrická a parametrická spojitost

2.2 Modelování křivek

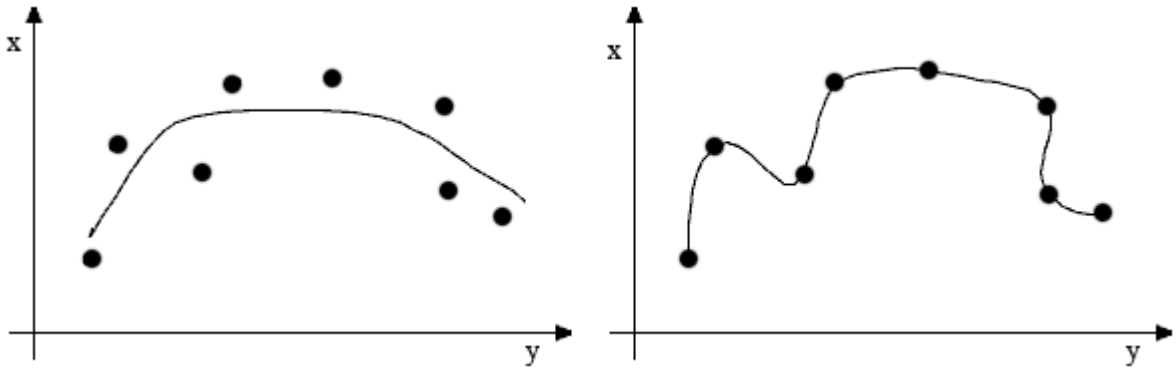
Základním druhem křivek parametrických, které se používají v počítačové grafice, jsou tzv křivky polynomiální. Bývají zapsány ve tvaru

$$Q_n(t) = a_0 + a_1 t + ... + a_n t^n$$

Křivky v tomto tvaru lze jednoduše vyčíslit. Z polynomiálních křivek lze skládat tzv. křivky po částech polynomiální. Jsou to křivky, jejichž jednotlivé segmenty jsou zároveň polynomiálními křivkami. V počítačové grafice se nejčastěji používají křivky třetího stupně. Nazývají se kubiky. Poskytují dostatečně širokou škálu tvarů, výpočet těchto křivek bývá nenáročný a lze s nimi snadno manipulovat. Výhodou je také možnost zaručit spojitost C^2, ta je často požadována při modelování v CAD systémech.

Modelování obvykle probíhá tak, že je zadáno několik řídících bodů, které dohromady tvoří tzv. řídící polygon a matematický aparát podle jejich polohy vypočte průběh křivky. Existují i metody, které umožňují zadávat křivku pomocí tečných vektorů, je možné zaručit spojitost a hladkost navázání aj.

Při modelování se můžeme setkat s dvojím přístupem ke křivkám. Jde o interpretaci úlohy řídících bodů. První je interpolace a druhá je aproximace. Interpolační křivky jsou generovány tak, že výsledná křivka prochází řídícími body, kdežto pro aproximační křivku obecně platí, že danými řídícími body procházet nemusí a její tvar je řízen rozestavením řídících bodů a použitou aproximační metodou (obr. 2.5). Jednotlivé metody těchto druhů křivek budou uvedeny v následujících kapitolách.



Obr. 2.5: Aproximační křivka (vlevo) a interpolační křivka

U interpolačních křivek se často vyskytuje pojem spline. Pochází z anglického jazyka a v češtině znamená pružné křivítko, pružný kovový pásek proložený body apod. Spline křivky jsou křivky po částech polynomiální. Důvodem jejich používání je minimalizace křivosti křivky a efektivní řízení

jejího tvaru. U interpolačních křivek mluvíme o přirozeném splinu. Tyto křivky mají spojitost C^{n-1}, kde *n* je stupeň křivky. U aproximačních křivek se hovoří o nepřirozeném splinu. O něm bude v této práci řeč v kapitole o uniformním kubickém B-splinu.

Na závěr této kapitoly je třeba ještě uvést často požadované vlastnosti křivek. První vlastností je invariance vůči lineárním transformacím a projekcím, která zaručuje, že například otočení řídícího polygonu a následné vygenerování křivky nám poskytne stejný výsledek jako otočení každého bodu z vygenerované křivky. Druhou je vlastnost konvexní obálky. Rozlišujeme silnou a slabou podmínku. Silná podmínka znamená, že jsou všechny body v konvexní obálce všech svých řídících bodů. Pro slabou podmínku platí, že část křivky leží v konvexní obálce a část mimo. Třetím požadavkem, který není ovšem vždy snadno splnitelný, je lokalita změn. Poslední často požadovanou vlastností je, aby křivka procházela krajními body. Tato vlastnost se hodí hlavně při navazování více křivek.

3 Interpolační křivky

Interpolační křivky jsou takové křivky, které prochází zadanými body. Tyto křivky jsou používány především v počítačové animaci pro definici dráhy pohybu. Jinak se v počítačové grafice příliš nepoužívají. V této kapitole jsou uvedeny nejčastěji používané metody pro výpočet těchto křivek.

3.1 Fergusonovy kubiky

Fergusonovy kubiky, často také označované jako Hermitovské kubiky, patří k nejčastěji používaným interpolačním křivkám. Křivky jsou určeny dvěma řídícími body P_0, P_1 a dvěma tečnými vektory $\vec{p}'_0$, $\vec{p}'_1$ v nich. Řídící body určují polohu křivky a zároveň jsou koncovými body křivky. Směr a velikost tečných vektorů určuje tvar a míru vyklenutí výsledné křivky. Čím je velikost vektoru větší, tím se k němu křivka více přimyká. Pokud jsou oba vektory nulové, tak se z křivky stane úsečka P_0P_1.

Obr. 3.1: Fergusonova kubika

Předpis pro výpočet Fergusonovy kubiky má v maticovém zápisu tvar

$$Q(t) = [t^3\, t^2\, t\, 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 2 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ \vec{p}'_0 \\ \vec{p}'_1 \end{bmatrix}.$$

Přepíšeme-li maticový tvar do jediné rovnice, dostaneme

$$Q(t) = P_0 F_1(t) + P_1 F_2(t) + \vec{p}'_0 F_3(t) + \vec{p}'_1 F_4(t),$$

kde F_1, F_2, F_3, F_4 jsou tzv. kubické Hermitovské polynomy ve tvaru

$$F_1(t) = 2t^3 - 3t^2 + 1,$$
$$F_2(t) = -2t^3 + 3t^2,$$
$$F_3(t) = t^3 - 2t^2 + t,$$
$$F_4(t) = t^3 - t^2.$$

Velká přednost těchto křivek se projeví při jejich navazování. Jelikož součástí definice těchto křivek jsou i tečné vektory v koncových bodech, lze jejich hladké navazování realizovat velmi snadno. Nevýhodou u těchto křivek je poměrně nesnadná editace tečného vektoru v prostoru, což je již ovšem mimo rámec této práce.

3.2 Křivky Kochanek-Bartels

Hlavní úlohou počítačových animací je interpolace bodů spojitou křivkou. Také by měla být poskytována určitá úroveň řízení v zadaných bodech. Vznikly proto křivky pojmenované podle svých autorů vycházející z Fergusonových kubik. Tyto křivky poskytují možnost řízení průběhu interpolovanými body. Konkrétně umožňují definovat napětí (tension), spojitost (continuity) a šikmost (bias). Z tohoto důvodu se také někdy nazývají TCB křivky.

Křivka je dána posloupností řídících bodů $P_0, P_1, .., P_n$ a řídících koeficientů a_i, b_i, c_i pro každý vnitřní řídící bod $P_1, P_2, ..., P_{n-1}$. Krajní body se podílejí na průběhu křivky, ale křivka nimi neprochází. Výpočet probíhá podle vztahů uvedených v kapitole 3.1. Pro výpočet je důležitá velikost tečného vektoru v každém řídícím bodě. Při výpočtu TCB křivek jsou pro řídící body definovány dva vektory, jenž se nazývají vstupní a výstupní vektor. Vstupní vektor je označen jako $\vec{l}_i$ a výstupní označujeme $\vec{r}_i$. Tyto vektory spočítáme ze vztahů

$$\vec{l}_i = \frac{(1-a)(1+b)(1-c)}{2}(P_i - P_{i-1}) + \frac{(1-a)(1-b)(1+c)}{2}(P_{i+1} - P_i),$$

$$\vec{r}_i = \frac{(1-a)(1+b)(1+c)}{2}(P_i - P_{i-1}) + \frac{(1-a)(1-b)(1-c)}{2}(P_{i+1} - P_i),$$

kde *a* je napětí, *b* je šikmost a *c* je spojitost.

Další informace o těchto křivkách se lze dozvědět v [1] v kapitole o počítačové animaci .

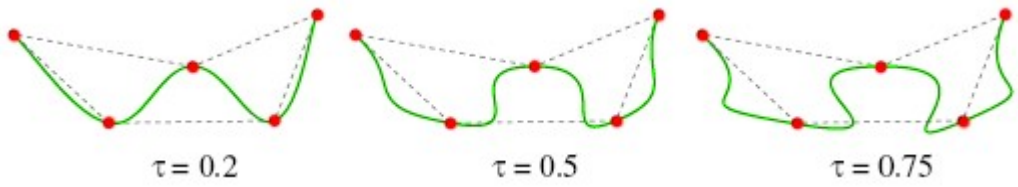
3.3 Kardinální spline a Catmull-Rom spline

Tyto křivky patří k dalším hojně využívaným interpolačním spline křivkám. Jejich využitím v počítačové grafice je například definice dráhy pohybu tělesa.

Obě křivky jsou definovány posloupností bodů $P_0, P_1, ..., P_n$. Křivka vychází z řídícího bodu P_1 a končí v bodě P_{n-1}, což znamená, že neinterpoluje krajními body. Pro výpočet těchto křivek můžeme použít vztah

$$Q(t) = [t^3\, t^2\, t\, 1] \begin{bmatrix} -\tau & 2-\tau & \tau-2 & \tau \\ 2\tau & \tau-3 & 3-2\tau & -\tau \\ -\tau & 0 & \tau & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{bmatrix},$$

kde τ je napětí, které ovlivňuje průběh křivky v okolí řídících bodů (viz obr. 3.2).



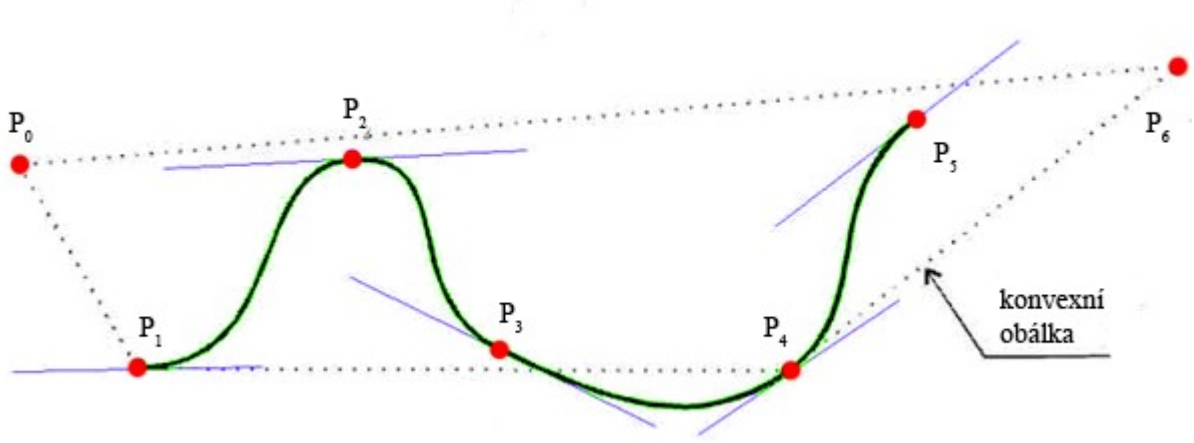
Obr. 3.2: Vliv napětí na tvar křivky

Catmull-Rom spline je speciální případ Kardinálního splinu, kdy napětí $\tau = 1/2$. Zároveň jsou tyto křivky speciálními případy křivek Kochanek-Bartels (více viz. [1, 3, 4]).

Výpočet těchto křivek probíhá nejprve pomocí bodů P_0, P_1, P_2, P_3 , poté P_1, P_2, P_3, P_4 a takto pokračuje až ke koncovému řídícímu bodu.

Toho, aby výsledná křivka procházela i koncovými řídícími body, dosáhneme zdvojnásobením těchto bodů. Křivka bude tedy dána body $P_0, P_1, \ldots, P_n$ a zároveň bude platit, že $P_0 = P_1$ a $P_n = P_{n-1}$.

Mezi vlastnosti těchto křivek patří například to, že tečný vektor bodu P_i je rovnoběžný s úsečkou spojující body P_{i-1}, P_{i+1} . Nevýhodou těchto křivek je, že výsledná křivka obecně neleží v konvexní obálce (obr. 3.3).



Obr. 3.3: Příklad křivky, kdy neleží v konvexní obálce

4 Aproximační křivky

Aproximační křivka je takové křivka, která obecně neprochází zadanými body, ale její průběh je pouze těmito body řízen. Způsob řízení je dán použitou aproximační metodou. Tyto křivky jsou v počítačové grafice hojně využívány. Pomocí těchto křivek lze namodelovat zřejmě jakýkoliv objekt. V této kapitole jsou uvedeny nejčastější aproximační křivky, které se používají v počítačové grafice.

4.1 Bézierovy křivky

Tyto křivky patří zřejmě k nejpopulárnějším aproximačním křivkám pro modelování v dvojrozměrné i trojrozměrné počítačové grafice. Často jsou využívány například pro definici vzhledu písma nebo v CAD systémech.

4.1.1 Obecné Bézierovy křivky

Tato větev počítačových křivek vznikla zobecněním Bézierových kubik. Bézierova křivka *n*-tého stupně vznikne zadáním $n+1$ bodů řídícího polygonu. Obecná Bézierova křivka je určena vztahem

$$Q(t) = \sum_{i=0}^{n} P_i B_i^n(t) ,$$

kde B_i^n jsou Bernsteinovy polynomy *n*-tého stupně

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}; \quad t \in \langle 0,1 \rangle; \quad i = 0,1,\ldots,n .$$

Z těchto vztahů lze snadno odvodit, že pokud položíme $t = 0$ a poté $t = 1$, křivka prochází prvním a posledním bodem řídícího polygonu. Po zderivování $Q(t)$ a dosazení do vzniklého vztahu týchž parametrů *t* dostaneme výrazy pro tečné vektory krajních bodů:

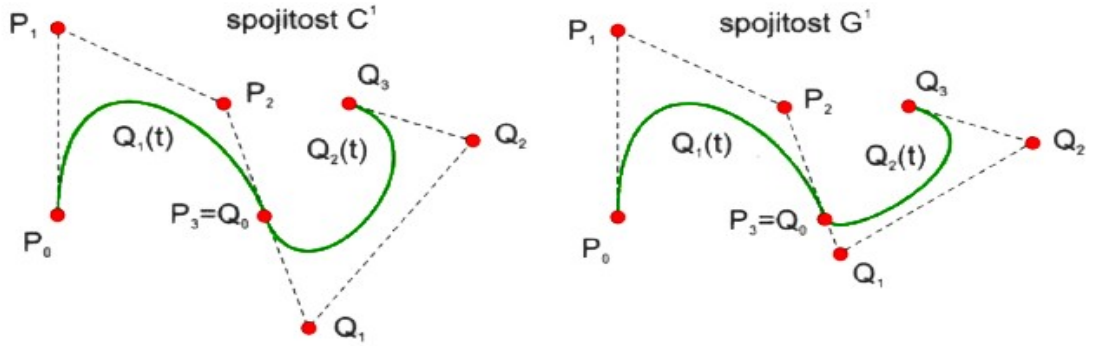
$$\vec{q}\,'(0) = n(P_1 - P_0) ,$$
$$\vec{q}\,'(1) = n(P_n - P_{n-1}) .$$

Tečné vektory mají tedy směr spojnice dvou krajních bodů a velikost *n*-násobku jejich velikosti, kde *n* je stupeň Bézierovy křivky.

Jednou z vlastností Bézierových křivek je, že při změně polohy jednoho jediného řídícího bodu dojde ke změně tvaru celé křivky. Toto chování není velmi často žádoucí, proto se křivky dělí na menší části a postupně se navazují. Nejčastěji jde o kubiky, ty budou popsány později. Tento postup se využívá také pro snížení výpočetní a tudíž i časové náročnosti výpočtu křivky.

Při navazování segmentů těchto křivek lze zaručit parametrickou spojitost C^0 identickou polohou koncového bodu prvního polygonu a počátečního bodu druhého polygonu. Hladké

parametrické navázání segmentů křivek, tedy spojitost C^1, zaručíme navíc, pokud je bod $P_n = Q_0$ a pokud je středem úsečky určené body P_{n-1} a Q_1 (viz obr. 4.1 vlevo). Geometrická spojitost G^1 se dá opět zaručit identitou koncového a počátečního bodu dvou řídících polygonů a navíc musí být splněna podmínka, že tyto koncové body musí ležet na úsečce, která spojuje předposlední řídící bod prvního segmentu s druhým řídícím bodem segmentu následujícího. Ovšem neplatí zde podmínka, že krajní body musí být umístěny ve středu této úsečky (obr. 4.1 vpravo).



Obr. 4.1: Spojení C^1 (vlevo) a G^1 (vpravo) dvou Bézierových kubik

Bernsteinovy polynomy B_i^n použité pro obecnou Bézierovu aproximaci mají následující vlastnosti:

1. $\forall i, n \in \mathbb{N} \cup \{0\}$ a $t \in \langle 0,1 \rangle$ je $B_i^n(t) \geq 0$
2. $\sum_{i=0}^{n} B_i^n(t) = 1$ pro $t \in \langle 0,1 \rangle$
3. $B_i^n(t) = (1-t)\, B_i^{n-1}(t) + tB_{i-1}^{n-1}(t)$

První vztah zaručuje nezápornost Bernsteinových polynomů. Společně s druhým pak zaručují to, že výsledná křivka leží vždy v konvexní obálce řídícího polygonu. Třetí vlastnost využil pro vytvoření rekurzivního algoritmu pro generování Bézierových křivek Paul de Casteljau. Tento algoritmus bude ještě v této práci popsán.

Při práci s Bézierovými křivkami se lze poměrně často setkat s požadavkem na editaci těch částí křivek, které to díky jejich stupni neumožňují. Jedním z řešení může být rozdělení řídícího polygonu na několik menších segmentů. Druhým řešením může být zvýšení stupně křivky. To se u Bézierových křivek provádí vytvořením nového řídícího polygonu. Tento má o jeden řídící bod více než původní, ovšem výsledným tvarem aproximované křivky se nijak neliší od té původní. Identitu obou křivek zajistí vztah

$$Q_i = \frac{i}{n+1} P_{i-1} + \left(1 - \frac{i}{n+1}\right) P_i, \quad i = 0,1,\ldots,n+1,$$

kde Q_i je řídící bod nového polygonu, P_i je řídící bod původního polygonu a *n* je stupeň křivky.

Na závěr je ještě nutné dodat, že jsou Bézierovy křivky invariantní vůči změně měřítka, rotaci a posunu.

4.1.2 Bézierovy kubiky

Tyto křivky jsou vůbec nejčastěji používanými. Jsou speciálním případem obecných Bézierových křivek. Na počátku 60. let je používal Pierre Bézier a díky své názornosti se staly velmi populární. Bézierova kubika je zadána čtyřmi řídícími body P_0, P_1, P_2, P_3. Křivka vychází z prvního bodu, končí v posledním bodě a vyklenutí řídí prostřední body. Je určena vztahem

$$Q(t) = \sum_{i=0}^{3} P_i B_i(t),$$

kde B_i jsou Bernsteinovy polynomy třetího stupně ve tvaru

$$B_0(t) = (1-t)^3,$$
$$B_1(t) = 3t(1-t)^2,$$
$$B_2(t) = 3t^2(1-t),$$
$$B_3(t) = t^3.$$

Maticový zápis pro výpočet Beziérovy kubiky je

$$Q(t) = [t^3\, t^2\, t\, 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}.$$

Ze vztahů pro určení tečných vektorů obecné Bézierovy křivky lze jednoduše odvodit vztahy pro Bézierovu kubiku. Mají tedy tvar:

$$\vec{q}\,'(0) = 3(P_1 - P_0) \text{ a } \vec{q}\,'(1) = 3(P_3 - P_2).$$

Jelikož jsou Bézierovy kubiky speciální podskupinou obecných Bézierových křivek, tak také pro tyto křivky platí pravidla spojitosti, invariance vůči lineárním transformacím atd.

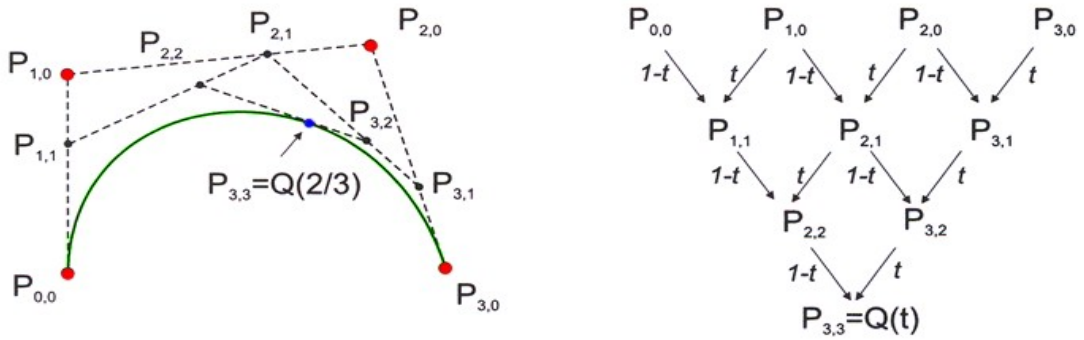
4.1.3 Algoritmus de Casteljau

Jak název vypovídá, jde o metodu, kterou vytvořil Paul de Casteljau. Jde o další možnou metodu, jak vypočítat body na Bézierových křivkách. Existují dva přístupy, jak pomocí tohoto algoritmu získat výslednou křivku.

Prvním je použití rekurzivního algoritmu de Casteljau, používané hlavně pro obecné Bézierovy křivky stupně *n*. Algoritmus vychází z rekurentní definice Bernsteinových polynomů. Vztah pro výpočet bodu $Q(t)$ má tvar

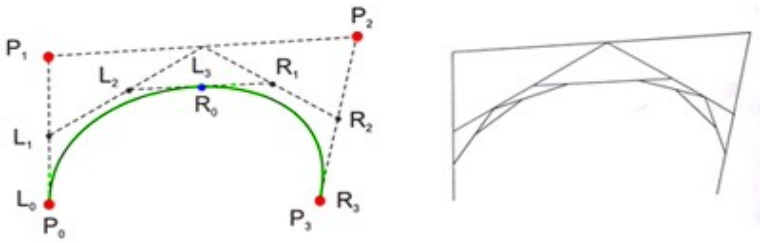
$$P_{j,i}(t) = (1-t)P_{j-1,i-1} + tP_{j,i-1},$$

kde $i = 1,2,\ldots,n$ a $j = i, i+1, \ldots, n$. Vstupem tohoto algoritmu jsou body řídícího polygonu a výpočet probíhá tak, že položíme $P_{i,0} = P_i$ a pomocí rekurentního vztahu postupně vypočteme hodnoty $P_{i,j}$ tak, jak je vidět na obrázku 4.2 vpravo. V posledním kroku je vypočtená hodnota rovna hodnotě bodu $Q(t)$ na křivce. Na obrázku 4.2 vlevo lze vidět výpočet bodu $Q(2/3)$.



Obr. 4.2: Algoritmus de Casteljau: výpočet bodu $Q(2/3)$ (vlevo) a schéma výpočtu (vpravo)

Druhým přístupem je rozdělení křivky na dvě části v libovolném místě. Toto místo je dáno zvolenou hodnotou parametru *t* pro $t \in \langle 0,1 \rangle$. Nejčastěji se používá varianta, kdy je křivka dělena v polovině, tzn. parametr $t = 1/2$. Tato metoda je často označována anglickým názvem „Divide and Conquer“. Mějme křivku n-tého stupně danou body $P_0, P_1, \ldots, P_n$. Postupným dělením získáme dvě skupiny řídících bodů $L_0, L_1, \ldots, L_n$ a $R_0, R_1, \ldots, R_n$. Ty určují příslušné části rozpůlené křivky a zároveň tvoří nově vzniklé polygony, které opět dělíme. Tento rekurzivní proces zastavíme až ve chvíli, kdy je například délka úsečky mezi dvěma sousedními body polygonu menší než úhlopříčka pixelu. Kritériem zastavení výpočtu může být také velikost plochy řídícího polygonu. Výhodou této metody je, že produkuje menší množství dat než jiné metody pro výpočet Bézierových křivek.



Obr. 4.3: Dělení Bézierovy křivky na dvě části (vlevo) a konvergence řídících bodů ke křivce

4.1.4 Racionální Bézierovy křivky

Jak napovídá název, jde o zobecnění Bézierových křivek. Při klasickém přístupu uživatel zadá body řídícího polygonu a program se postará buď o výpočet bodů křivky daného řádu, nebo o výpočet bodů na křivkách nižšího řádu. Pokud chce poté editovat její tvar, musí změnit polohu některého z řídících bodů, což ovšem často nevede k požadovanému výsledku. Proto tato metoda přiřadí každému bodu řídícího polygonu, kromě jeho souřadnic, ještě reálné číslo, na jehož hodnotě tvar křivky také závisí.

Racionální Bézierova křivka je určena posloupností bodů $P_0, P_1, \ldots, P_n$ řídícího polygonu a odpovídající posloupností reálných čísel $\omega_0, \omega_1, \ldots, \omega_n$, jež se nazývají váhové koeficienty. Bézierova racionální křivka řádu *n* je potom určena vztahem

$$Q(t) = \frac{\sum_{i=0}^{n} \omega_i P_i B_i^n(t)}{\sum_{i=0}^{n} \omega_i B_i^n(t)} = \sum_{i=0}^{n} P_i R_i^n(t),$$

kde $i = 0,1,\ldots,n \in \mathbb{N}$, $\omega_i \geq 0 \in \mathbb{R}$ a

$$R_i^n(t) = \frac{\omega_i B_i^n(t)}{\sum_{i=0}^{n} \omega_i B_i^n(t)}.$$

B_i^n jsou Bernsteinovy polynomy a R_i^n jsou racionální Bernsteinovy polynomy. Pro racionální Bézierovy křivky platí:

1. $\forall i, n \in \mathbb{N} \cup \{0\}$ a $t \in \langle 0,1 \rangle$ je $R_i^n(t) \geq 0$
2. $\sum_{i=0}^{n} R_i^n(t) = 1$ pro $t \in \langle 0,1 \rangle$
3. Je-li $\omega_i = 1$, $i = 0,1,\ldots,n$, pak je $R_i^n = B_i^n$

Z posledních dvou vztahů lze vyvodit, že jsou racionální Bézierovy křivky opravdu zobecněním Bézierových křivek. Ty neracionální můžeme označit jako speciální případ racionálních, kdy se váhové koeficienty všech bodů rovnají jedné.

Dosazením krajních hodnot parametru *t* do rovnice pro výpočet křivky zjistíme, že křivka, stejně jako u neracionální formy Bézierových křivek, prochází koncovými body. Vztahy pro tečny v krajních bodech mohou být vyjádřeny takto:

$$\vec{q}\,'(0) = \frac{\omega_1 n (P_1 - P_0)}{\omega_0},$$

$$\vec{q}\,'(1) = \frac{\omega_{n-1} n (P_n - P_{n-1})}{\omega_n}.$$

Pokud opět dosadíme za váhové koeficienty jedničku, získáme vztahy pro tečny obecných Bézierových křivek. Výsledná aproximační křivka leží v konvexní obálce řídícího polygonu, jsou-li hodnoty všech váhových koeficientů nezáporné. Podobně, jako u neracionálních Bézierových křivek, lze zvýšit i u těchto křivek stupeň křivky přidáním nového bodu do řídícího polygonu. Mějme původní křivku, která je zadána řídícími body $P_0, P_1, \ldots, P_n$ a váhovými koeficienty $\omega_0, \omega_1, \ldots, \omega_n$ a novou křivku určenou body a jejich váhami $Q_0, Q_1, \ldots, Q_{n+1}$, $\omega_{0,} \omega_{1,} \ldots, \omega_{n+1}$. Aby byl tvar křivky zachován musí pro nové řídící body platit vztah

$$Q_i = \eta_i \frac{\omega_{i-1}}{\overline{\omega_i}} P_{i-1} + (1 - \eta_i) P_i ,$$

kde

$$\eta_i = \frac{i}{i + n}, \quad \overline{\omega_i} = (1 - t)\omega_{i-1} + t\,\omega_i .$$

Jak již bylo uvedeno u obecných Bézierových křivek, výpočet může být často náročný, proto se i racionální Bézierovy křivky v praxi dělí na menší segmenty. Při navazování jednotlivých částí křivky nás tedy také zajímá, jak zajistit spojitý, popřípadě hladký průběh křivky. Pro spojité navázání platí opět, že koncový bod prvního řídícího polygonu musí být totožný s počátečním bodem navazujícího polygonu. Pro hladké spojení dvou segmentů musí navíc platit:

$$\omega_n = \eta_0 \text{ a } \omega_{n-1}(P_n - P_{n-1}) = \eta_1(Q_1 - Q_0) ,$$

kde P_i a ω_i jsou řídící body a jejich váhové koeficienty prvního polygonu a Q_i, η_i jsou řídící body a jejich váhové koeficienty navazovaného segmentu křivky.

Racionální křivky jsou obecně pro počítačovou grafiku velkým přínosem, protože umožňují změnu tvaru křivky beze změny polohy řídícího bodu. Pomocí racionálních Bézierových křivek je možné vytvářet kuželosečky. Další informace o těchto křivkách, například výpočet pomocí racionálního algoritmu de Casteljau, lze najít v [5].

4.2 Coonsovy kubiky

Coonsova kubika zaujímá vedle Bézierových křivek významné postavení. Tuto metodu navrhl, jak již název naznačuje, S. A. Coons a díky svým dobrým geometrickým vlastnostem je hojně využívána. Coonsova kubika velmi přispěla do teorie spline křivek.

Coonsova kubika je určena stejně jako Bézierova kubika čtyřmi řídícími body P_0, P_1, P_2, P_3.

Výpočet je dán vztahem

$$Q(t) = \frac{1}{6} \sum_{i=0}^{3} P_i C_i(t) ,$$

kde C_i jsou kubické polynomy ve tvaru

$$C_0(t) = (1-t)^3,$$
$$C_1(t) = 3t^3 - 6t^2 + 4,$$
$$C_2(t) = -3t^3 + 3t^2 + 3t + 1,$$
$$C_3(t) = t^3.$$

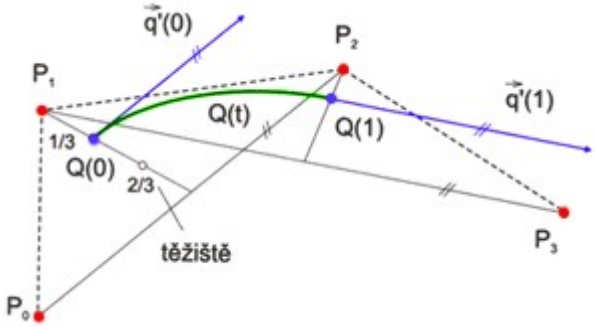
Maticový zápis pro výpočet Coonsovy kubiky je

$$Q(t) = \frac{1}{6}[t^3\, t^2\, t\, 1]\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}\begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}.$$

Dosazením krajních hodnot *t* do vztahu pro výpočet bodu na křivce dostáváme rovnice pro výpočet krajních bodů křivky. Mají tvar:

$$Q(0) = \frac{P_0 + 4P_1 + P_2}{6} \text{ a } Q(1) = \frac{P_1 + 4P_2 + P_3}{6}.$$

Jak lze vidět, tak výsledná křivka generovaná touto metodou nezačíná a nekončí v koncových bodech řídícího polygonu. Počáteční bod leží v tzv. antitěžišti trojúhelníka P_0, P_1, P_2 a koncový bod zase v antitěžišti trojúhelníka P_1, P_2, P_3 (viz obr. 4.4).



Obr. 4.4: Coonsova kubika

Po derivaci vztahu pro výpočet křivky a dosazení krajních hodnot parametru *t* dostaneme vztahy pro tečné vektory v krajních bodech:

$$\vec{q}\,'(0) = \frac{P_2 - P_0}{2}, \quad \vec{q}\,'(1) = \frac{P_3 - P_1}{2}.$$

Vektory druhých derivací mají podobu:

$$\vec{q}\,''(0) = P_0 - 2P_1 + P_2, \quad \vec{q}\,''(1) = P_1 - 2P_2 + 2P_3$$

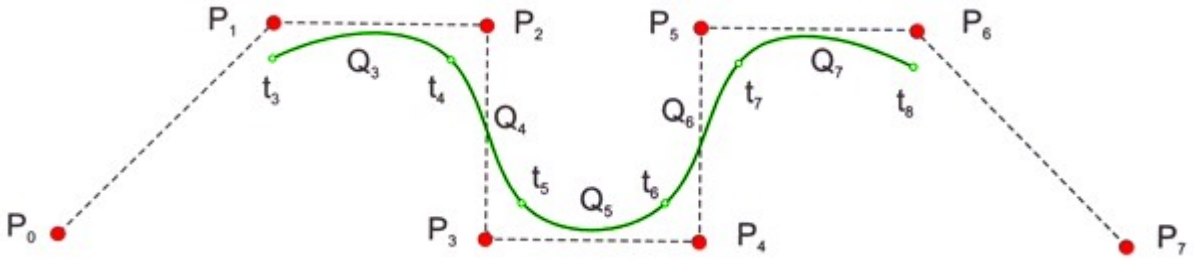
Velký význam má u Coonsových kubik násobnost bodů řídícího polygonu. Pokud položíme $P_0 = P_1$, vytvoříme tzv. dvojnásobný bod a trojúhelník tvořený body P_0, P_1, P_2 zdegenerujeme

na úsečku P_0, P_2 a křivka začíná v jedné šestině této úsečky. Položíme-li $P_0 = P_1 = P_2$, vytvoříme tím tzv. trojnásobný bod. Křivka bude začínat v tomto bodě a končit v jedné šestině úsečky P_0, P_3.

4.3 Uniformní kubický B-spline

B-spline křivky patří do skupiny nepřirozených spline křivek. To proto, že se nejedná o křivky interpolační, ale o křivky aproximační (křivka neprochází řídícími body).

Uniformní kubický B-spline je též nazýván jako Coonsův kubický B-spline. Vzniká navázáním Coonsových kubik. Následující segment je vždy dán posledními třemi body segmentu předchozího a jedním bodem následujícího.



Obr. 4.5: Coonsův kubický B-spline

B-spline je oproti Coonsově kubice určen $n \geq 4$ body a skládá se z $n-3$ segmentů. Coonsova kubika je tedy sama o sobě B-spline křivkou.

Pokud porovnáme tečné vektory a vektory druhých derivací po sobě jdoucích segmentů, zjistíme, že jsou tyto vektory identické. Křivka je tedy v koncových uzlech C^2 spojitá.

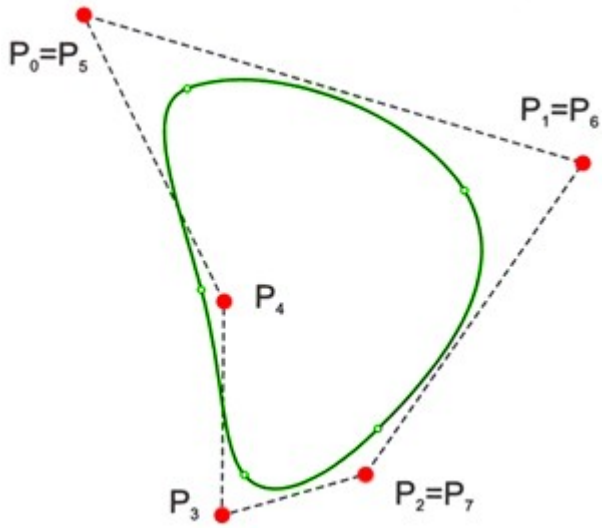
Coonsův kubický B-spline je generován $n+1$ body řídícího polygonu $P_0, P_1, \ldots P_n$ a je složen z $n-2$ segmentů $Q_1, Q_2, \ldots, Q_n$ (obr. 4.5). Parametr *t* probíhá interval $\langle t_3, t_{n+1} \rangle$ a hodnoty tohoto parametru definují uzlový vektor. Hodnoty mají konstantní vzdálenost (ve smyslu parametru *t*), a proto vektor označujeme jako uniformní.

V praxi se lze setkat s uzavřeným Coonsovým B-splinem (obr. 4.6). Ten je vytvořen zopakováním prvních tří bodů na jeho konci.

Mezi důležité vlastnosti B-spline křivek patří invariance vůči lineárním transformacím. Výsledná křivka leží v konvexní obálce svého polygonu. To platí i pro jednotlivé segmenty křivky.

Podobně jako u Coonsových kubik, má i u těchto křivek význam násobnost bodů. Pokud ztrojnásobíme krajní body, docílíme toho, že bude křivka procházet krajními body. Pokud budeme znásobovat body uvnitř křivky, dosáhneme různých deformací této křivky.

Změna polohy jednoho bodu této křivky vždy změní tvar čtyř segmentů této křivky.



Obr. 4.6: Uzavřený Coonsův B-spline

4.4 NURBS

Neuniformní racionální B-spline křivky (angl. non uniform rational B-spline) jsou dvojím zobecněním B-spline křivek uvedených v předchozí části. Termín neuniformní je zde odvozen od vzdálenosti mezi uzly ve smyslu parametru *t*. Ta u těchto křivek nemusí být konstantní. Racionalita znamená, podobně jako u Bézierových křivek, že tvar křivky ovlivňuje nejen poloha, ale i váhový koenficient.

Křivka NURBS je určena $n+1$ body P_i řídícího polygonu, kde $i = 0,\ldots,n$. Dále je určena řádem B-spline k (nejvyšší stupeň je $k-1$) a uzlovým vektorem U délky $n+k+1$, který je dán posloupností neklesajících reálných čísel, tzv. uzlových hodnot $t_0 \le t_1 \le \ldots \le t_{n+k}$. Příkladem uzlového vektoru s uniformním rozložením může být $U_1=\{-2,-1,0,1,2,3\}$ a příklad tzv. okrajového uzlového vektoru může vypadat takto: $U_2=\{0,0,0,1,1,1\}$.

Křivka NURBS je určena vztahem

$$Q(t) = \frac{\sum_{i=0}^{n} \omega_i P_i N_{i,k}(t)}{\sum_{i=0}^{n} \omega_i N_{i,k}(t)},$$

kde ω_i je váha *i*-tého bodu řídícího polygonu a $N_{i,k}(t)$ jsou tzv. normalizované B-spline bázové funkce definované rekurentním vztahem

$$N_{i,1}(t) = \begin{cases} 1 & pro\, t_i \le t < t_{i+1} \\ 0 & jinde \end{cases}$$

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t) \text{ pro } t_i < t_{i+1+k},\ 0 \le i \le n .$$

Během výpočtu se mohou vyskytnout i výrazy, které mají ve jmenovateli nulu. Hodnota takového výrazu je pak rovna nule. Někdy se lze setkat se zápisem, který využívá racionální B-spline báze

$$R_{i,k} = \frac{\omega_i N_{i,k}(t)}{\sum_{j=0}^{n} \omega_j N_{j,k}(t)} .$$

Křivku NURBS lze poté přepsat jednodušeji

$$Q(t) = \sum_{i=0}^{n} P_i R_{i,k}(t) .$$

Polynomy NURBS báze N_i^n mají následující vlastnosti:

1. $\forall i, n \in \mathbb{N} \cup \{0\}$ a $t \in \langle 0,1 \rangle$ je $N_i^n(t) \ge 0$
2. $\sum_{i=0}^{n} R_i^n(t) = 1$ pro $t \in \langle 0,1 \rangle$
3. $N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t)$ pro $t_i < t_{i+1+k},\ 0 \le i \le n$

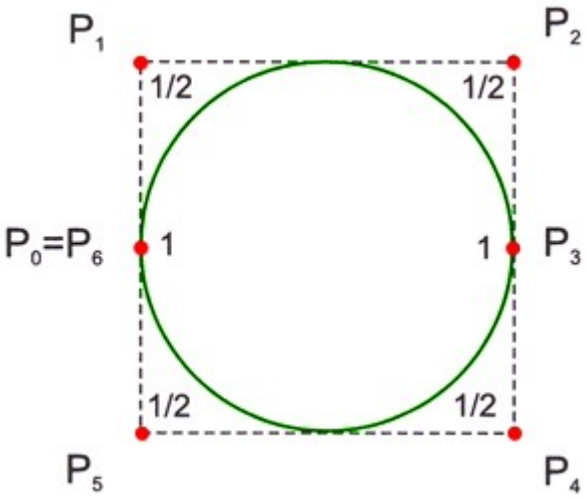
První vztah zaručuje nezápornost polynomů B-spline báze. První a druhý zaručují, že výsledná křivka vždy leží v konvexní obálce řídícího polygonu. Třetí vztah je rekurentní definicí bázového polynomu řádu *k* pomocí lineární kombinace dvou po sobě následujících bázových polynomů řádu $k-1$. Tato vlastnost se stala základem de Boorova algoritmu pro výpočet bodu na křivce. O této technice se lze dozvědět více např. v [5].

Jedním z nejdůležitějších prostředků, které křivky NURBS poskytují při modelování, je možnost vložení nových uzlových bodů bez nutnosti zvyšování řádu křivky. Nově vložené uzlové body pak umožňují společně s příslušně pozměněnými řídícími body vymezit ty části křivky, které jsou potřeba lokálně ovlivnit. Další vlastností NURBS křivek je to, že leží v konvexní obálce svého řídícího polygonu. Stejně tak leží v konvexní obálce svého řídícího polygonu i každý jednotlivý segment této křivky. Změna váhy jednoho řídícího bodu má vliv jen na část křivky. NURBS křivky jsou invariantní vůči transformacím a vůči paralelní a perspektivní projekci. Lze pomocí správně nastavených váhových koeficientů a uzlových bodů vyjádřit kuželosečky (obr. 4.7).

Okrajový uzlový vektor

$$U = \{\underbrace{\alpha, \alpha, \ldots, \alpha}_{k}, \underbrace{t_k, \ldots, t_{n-k}}_{n+k+1}, \underbrace{\beta, \beta, \ldots, \beta}_{k}\}$$

zajistí, že křivka prochází koncovými body řídícího polygonu.



Obr. 4.7 : Kružnice vytvořená pomocí křivky NURBS. Váhové koeficienty lze vidět na obrázku a uzlový vektor má tvar $U=\{0,0,0,1/4,1/2,1/2,3/4,1,1,1\}$

Pro uzlový vektor

$$U=\{\underbrace{0,0,...,0}_{k},\underbrace{1,1,...,1}_{k}\}$$

jsou normalizované B-spline báze rovny Bernsteinovým polynomům stupně k a křivka NURBS se tak stává racionální Bézierovou křivkou. Pokud se všechny váhové koeficienty rovnají jedné, je NURBS neracionální Bézierovou křivkou.

5 Praktická realizace

Nedílnou součástí této bakalářské práce je programový výstup ve formě demonstrační aplikace určené pro výuku vybraných dvojrozměrných křivek. Vývoj této aplikace probíhal v několika fázích. První fází byl výběr programovacích prostředků k vytvoření výsledného programu. Další fází byl vývoj uživatelského rozhraní, který probíhal v různé míře po celou dobu vývoje aplikace paralelně s ostatními fázemi. Mezi ně patří zejména návrh datových struktur, implementace algoritmů pro výpočet křivek, implementace funkcí pro vykreslování křivek a manipulaci s nimi. Poté byla aplikace obohacena o požadovanou možnost krokování výpočtu a byl doplněn výpočet konvexní obálky dle Jarvisova algoritmu (viz. [6]). Nakonec byly doplněny texty s teorií a funkce, které přímo neovlivňují běh aplikace, ale mohou uživateli zpříjemnit práci (např. nastavování barev jednotlivých elementu nebo nápověda).

5.1 Výběr programovacích prostředků

Pro vývoj aplikace jsem zvolil vývojové prostředí MS Visual Studio 2005. Zvoleným implementačním jazykem je C# .NET 2.0 podporovaný knihovnami Tao Framework 2.0.

MS Visual Studio 2005 je dle mého názoru velmi příjemné vývojové prostředí, které umožňuje jednoduše vytvářet aplikace. Podporuje vývoj v několika programovacích jazycích. Při práci na aplikaci jsem ocenil zejména nástroj s názvem Intellisence, který dokáže při psaní kódu zkoumat tento kód a nabízet možné varianty pro editaci. Dále jsem také ocenil Debugger, který sice občas havaroval, ale i přesto mi pomohl odhalit a odladit několik fatálních chyb, jež nebyly při vizuální kontrole kódu vidět.

Platforma .NET Framework je podobná platformě J2EE. Funguje následujícím způsobem: zdrojový kód napsaný v některém z podporovaných jazyků je zkompilován do tzv. mezikódu (MSIL – Microsoft Intermediate Language) a až na výsledné architektuře se přeloží do strojového kódu. Výhodou .NETu je, že odstraňuje např. problémy s nekompatibilitou dynamických knihoven. Nevýhodou například náročnější a pomalejší programy.

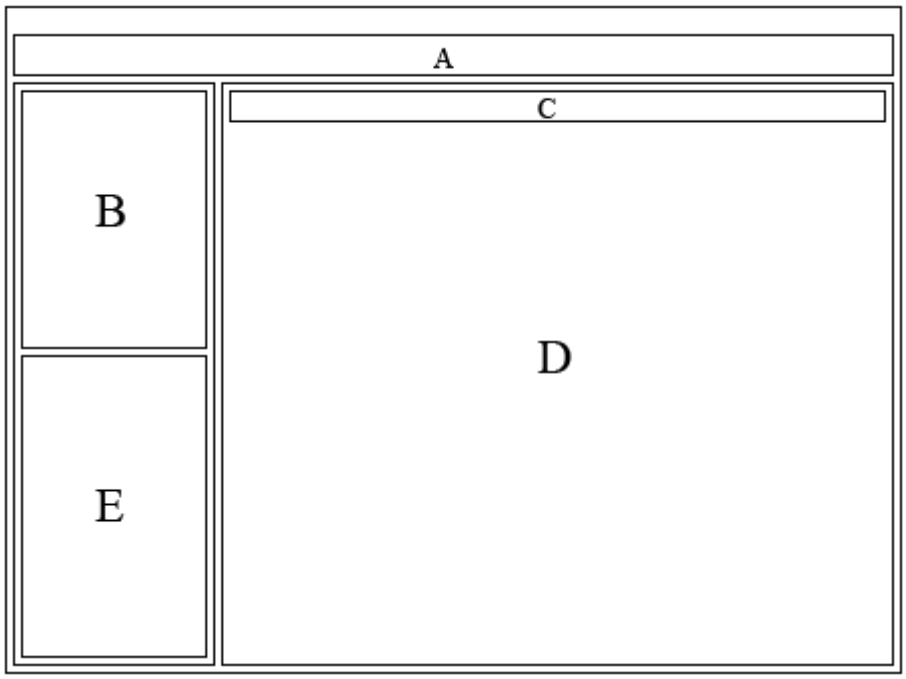
Jazyk C# (viz. [7]) je vysoko úrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft společně s platformou .NET Framework. Tento jazyk je založen na jazycích C++ a Java. Pomocí tohoto jazyka lze vytvářet konzolové aplikace, formulářové aplikace a dokonce i například webové aplikace. Vybral jsem si jej, protože jsem si chtěl prohloubit znalosti o tomto jazyku a věřím, že má do budoucna velký potenciál.

Tao Framework (www.taoframework.com) je balík grafických a herních knihoven pro .NET platformu a její alternativní open-source implementaci Mono. Tento balík mi umožnil pracovat velmi jednoduše s OpenGL.

5.2 Návrh uživatelského rozhraní

S rozvojem informačních technologií došlo také k rozvoji uživatelských rozhraní. Doba, kdy byl vrcholem nevzhledný textový editor nabízející několik málo funkcí, je již dávno pryč. V dnešní době jsou aplikace často „přeplácané“ funkcemi, proto je nutné, aby tvůrci těchto rozhraní pečlivě uvážili rozmístění ovládacích prvků, jejich velikosti apod. Aplikace by měly nabízet intuitivní ovládání, aby práce v nich byla efektivní a pro uživatele příjemná.

Jak jsem se již zmínil, vývoj uživatelského rozhraní probíhal téměř po celou dobu vývoje aplikace a návrhů bylo hned několik. Při vývoji jsem kladl důraz především na velikost pracovní plochy, protože ta je u aplikace demonstračního rázu velmi důležitá. Nakonec jsem zvolil to uživatelské rozhraní, ve kterém se mi pracovalo nejlépe a jsem přesvědčen o tom, že jej ocení i ostatní uživatelé. Schéma uživatelského rozhraní lze vidět na obrázku 5.1.



Obr. 5.1: Schéma uživatelského rozhraní

A – hlavní menu

B – menu s kapitolami jednotlivých křivek

C – záložky pro přepínání mezi teorií a kreslícím plátnem

D – texty s teorií / kreslící plátno
E – ovládací panel

Po zapnutí aplikace nejsou části C a E viditelné a v části D je umístěn pouze obrázek. Po kliknutí na první položku menu je zobrazen v části D text s úvodní teorií do dvojrozměrných křivek. Části C a E jsou neviditelné. Pokud uživatel klikne na některou položku s konkretní křivkou, v části D se zobrazí teorie k dané křivce a části C a E se zviditelní. Ovládací panel (E) se ovšem zobrazí v pasivním módu. Aktivuje se teprve až po kliknutí na položku pro ukázku v části C. Toto chování jsem zvolil z důvodu, aby měl uživatel stále v podvědomí fakt, že si může po přečtení teorie vyzkoušet také praxi. Po změně kapitoly se vždy objeví v části D teorie dané křivky a nástrojová lišta je opět pasivována. Jelikož v okně aplikace s minimálním rozlišením nebylo pro ovládací panel příliš mnoho místa pro doplňující funkce, rozhodl jsem se přidat k aplikaci další formulář, který je zobrazován uživatelem volitelně.

5.3 Důležité datové struktury

Ještě před samotnou implementací výpočtu křivek bylo nutné navrhnout datovou strukturu, ve které bych uchovával informace o řídících bodech a o bodech ležících na křivce. Dále pak bylo nutné navrhnout datovou strukturu pro uložení barvy obrazového bodu (pixelu).

První datovou strukturou je třída *CanvasPoint*. Pomocí této třídy lze reprezentovat řídící body i body na křivce. Pro tento účel je tato třída využita v kombinaci s dynamickým polem (v použitém programovacím jazyce – *List<CanvasPoint>*). Třída obsahuje dva konstruktory pro řídící body neracionálních křivek a vypočtené body a pro řídící body racionálních křivek. Dále obsahuje tzv. vlastnosti pro získání a nastavování jednotlivých proměnných třídy.

```
class CanvasPoint
{
        double x;               // x-ova souradnice bodu
        double y;               // y-ova souradnice bodu
        double weight; // vaha bodu (u ridicich bodu racionalni krivky)
        // konstruktor pro neracionalni krivku a body na krivce
        public CanvasPoint(double x, double y)
        {
                this.x = x;
                this.y = y;
                this.weight = 1;
        }
```

```
        // konstruktor pro racionalni krivku
        public CanvasPoint(double x, double y, double weight)
        {
                this.x = x;
                this.y = y;
                this.weight = weight;
        }
        public X
        {
                get { return this.x; }
                set { this.x = value; }
        }
        public Y
        {
                get { return this.y; }
                set { this.y = value; }
        }
        public Weight
        {
                get { return this.weight; }
                set { this.weight = value; }
        }
}
```

Druhou datovou strukturou je struktura *PixelColor*. Reprezentuje barvu pixelu v formátu RGB a v programu je využita hlavně jako dvojrozměrné pole reprezentující kreslící plátno. Struktura obsahuje konstruktor a tzv. vlastnosti pro získání jednotlivých složek barvy.

```
Public struct PixelColor
{
        private byte red;               // cervena slozka
        private byte green;     // zelena slozka
        private byte blue;              // modra slozka
        // konstruktor
        public PixelColor(byte red, byte green, byte blue)
        {
                this.red = red;
                this.green = green;
                this.blue = blue;
        }
```

```
        public Red
        {
                get { return this.red; }
        }

        public Green
        {
                get { return this.green; }
        }
        public Blue
        {
                get { return this.blue; }
        }
}
```

5.4 Implementace křivek

Po prostudování jednotlivých algoritmů pro výpočet 2D křivek používaných v počítačové grafice přišlo na řadu rozhodování, které křivky bude moje aplikace demonstrovat. Aby se aplikace nezaměřovala pouze na jeden jediný druh křivek ve smyslu interpretace řídících bodů, zvolil jsem Kardinální spline jako zástupce interpolačních křivek a dále pak Bézierovu kubiku, algoritmus de Casteljau pro výpočet Bézierovy křivky, Coonsovu kubiku a křivku NURBS.

Kardinální spline jsem si vybral, protože lze pomocí koeficientu přilnavosti jednoduše demonstrovat změnu tvaru křivky, dále pak lze demonstrovat chování tečen křivky v řídících bodech a také případy, kdy křivka neleží v konvexní obálce.

Na Bézierově kubice lze názorně demonstrovat nejen parametrickou spojitost C^0 a C^1, ale také geometrickou spojitost G^0 a G^1. Dále lze například demonstrovat, jak se vytváří pomocí Bézierovy kubiky kružnice. Algoritmus de Casteljau jsem vybral hlavně pro názornost výpočtu při krokování.

Coonsovu kubiku jsem zvolil pro účel demonstrace násobnosti řídících bodů a zopakování řídících bodů.

A nakonec křivka NURBS. Tuto křivku jsem si vybral, protože pochopení těchto křivek není úplnou samozřejmostí a ze zkušeností vím, že se tomu i plno lidí brání. Tyto křivky umožňují obrovské možnosti. Pomocí změn váhových koeficientů řídících bodů a uzlového vektoru lze dosáhnout všech možných tvarů. Lze například namodelovat kružnici.

Výpočet bodů ležících na těchto zvolených křivkách byl implementován podle výpočetních vztahů uvedených v předchozích částech této práce. Jednotlivé metody pro výpočet křivek lze nalézt ve složce se zdrojovými kódy v souboru *CalcCurves.cs* v třídě *CalcCurves*.

5.5 Vykreslování plátna

Jelikož je plátno reprezentováno dvojrozměrným polem pixelů bylo nutné implementovat funkce, které se postarají o obarvení každého pixelu správnou barvou. Pro tyto účely jsem vytvořil třídu *InsertElements* obsahující kolekci statických metod. Za zmínku stojí například metoda *InsertLine*, která se postará o vykreslení úsečky pomocí Bresenhamova algoritmu (viz. [2]).

K vykreslení tohoto pole na obrazovku pak stačí komponenta *SimpleOpenGLControl* a volání několika OpenGL funkcí zprostředkovaných již zmiňovaným balíkem Tao Framework. Vykreslování plátna je závislé na události OnPaint. Pokud dojde k této události a pokud je vykreslení nutné, tak se tak stane.

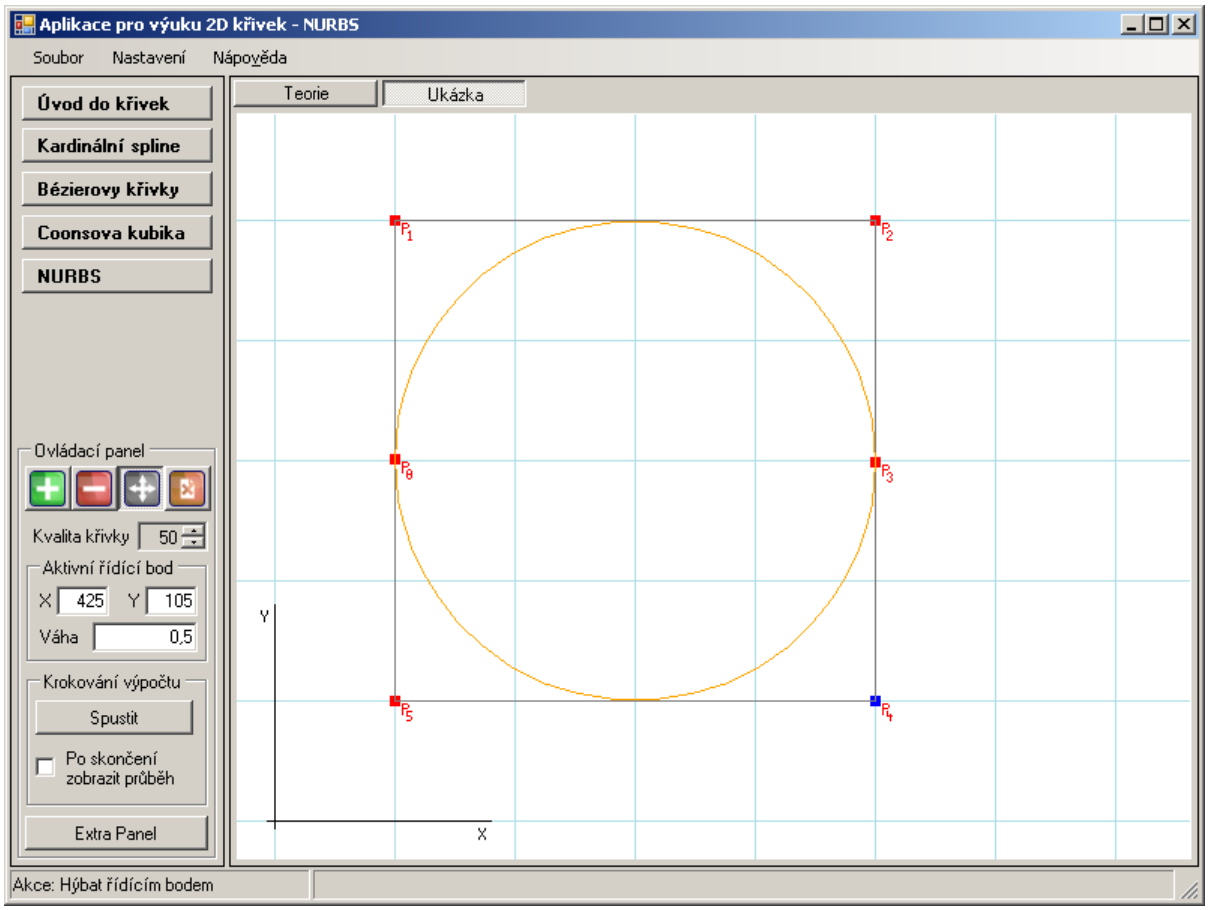
5.6 Krokování výpočtu

Krokování výpočtu křivek je dle zadání nedílnou součástí aplikace. Bylo implementováno dvěma způsoby a uživatel si může vybrat, který způsob mu více vyhovuje.

První implementovaná varianta je dostupná ze základního ovládacího panelu umístěného pod tlačítky s kapitolami. Tato varianta je implementována pomocí časovače (komponenta Timer), který v pevně daných časových úsecích volá funkci pro výpočet křivky v krokovacím módu s aktuálním parametrem *t*.

Druhá varianta je dostupná z rozšířené nástrojové lišty. Tu jsem implementoval opět pomocí časovače, ovšem s tím rozdílem, že lze nastavovat časový interval, a také pomocí tlačítek pro manuální krokování výpočtu pro daný parametr *t*.

U obou těchto variant lze výsledek vidět na plátně v podobě bodu, který se pohybuje po křivce a také lze nalézt v textovém souboru s názvem *curve.log* záznam se souřadnicemi bodů ležících na křivce pro jednotlivé hodnoty parametru *t*.



Obr. 5.2 Ukázka výsledné aplikace

6 Závěr

V této bakalářské práci jsem se zabýval metodami pro výpočet a modelování dvojrozměrných počítačových křivek a jejich následným převedením do praxe. Práce na tomto projektu byla velmi zajímavá. Oprášil jsem znalosti získané v předmětu Základy počítačové grafiky, který je vyučován v druhém ročníku bakalářského studia. Zároveň jsem si také rozšířil obzory v tomto odvětví počítačové grafiky a pochopil některé věci, které mi předtím nebyly zcela jasné. Velký přínos má pro mne také hlubší seznámení s programovacím jazykem C# .NET. Díky jeho vzrůstající oblibě se s ním určitě nesetkávám naposledy.

Dle mého názoru lze demonstrační aplikaci, kterou jsem vytvořil, považovat za plnohodnotnou učební pomůcku. Ocení ji jak studenti, tak i pedagogové. Své příznivce by také mohla získat mezi zájemci o počítačovou grafiku z řad široké veřejnosti.

Ačkoliv je aplikace pro danou problematiku a pro vybrané křivky plně funkční, vždy se dá něco vylepšit. V případě této aplikace by se mohlo jednat o optimalizaci použitých metod a algoritmů jak pro výpočet křivek, tak pro zobrazování výsledků. Aplikace totiž při těchto operacích, obzvláště při pohybu řídícího bodu, vykazuje zvýšené nároky na výkon počítače. Druhým způsobem, jak tuto aplikaci rozšířit, by mohla být možnost rozšiřování okruhu křivek formou zásuvných modulů. Aplikace by se také nemusela soustředit pouze na křivky v rovině, ale mohla by umožnit demonstrovat křivky v prostoru. To už je ovšem téma, které přesahuje rámec této práce.

Literatura

[1] Žára, J., Beneš, B., Sochor, J., Felkel, P.: Moderní počítačová grafika. 2. vydání, Brno, Computer Press, 2004.

[2] Sochor, J., Žára, J.: Algoritmy počítačové grafiky. VŠ skripta, 1. vydání, Praha, Vydavatelství ČVUT, 1993.

[3] Twigg, C.: Catmull-Rom Splines. [online], 2003, [cit. 2008-07-20].
URL <http://www.cs.cmu.edu/~fp/courses/graphics/asst5/catmullRom.pdf>

[4] Armstrong, J.: Catmull-Rom Splines. [online], 2006, [cit. 2008-07-20].
URL <http://www.algorithmist.net/media/catmullrom.swf>

[5] Marsh, D.: Applied Geometry for Computer Graphics and CAD. 2. vydání, Springer, 2000.

[6] Felkel, P.: Výpočetní geometrie – Datové struktury a algoritmy. [online], [cit. 2008-07-20]
URL <http://cs.felk.cvut.cz/~berezovs/dsa/praco/Lesson11pub.pdf>

[7] Nagel, Ch.: C# 2005 – Programujeme profesionálně. Brno, Computer Press, 2007

Seznam příloh

Příloha 1. CD obsahující elektronickou verzi technické zprávy, zdrojové kódy aplikace a manuál k aplikaci. Dále obsahuje spustitelnou verzi aplikace a instalační soubory .NET Framework 2.0 a Tao Framework 2.0.